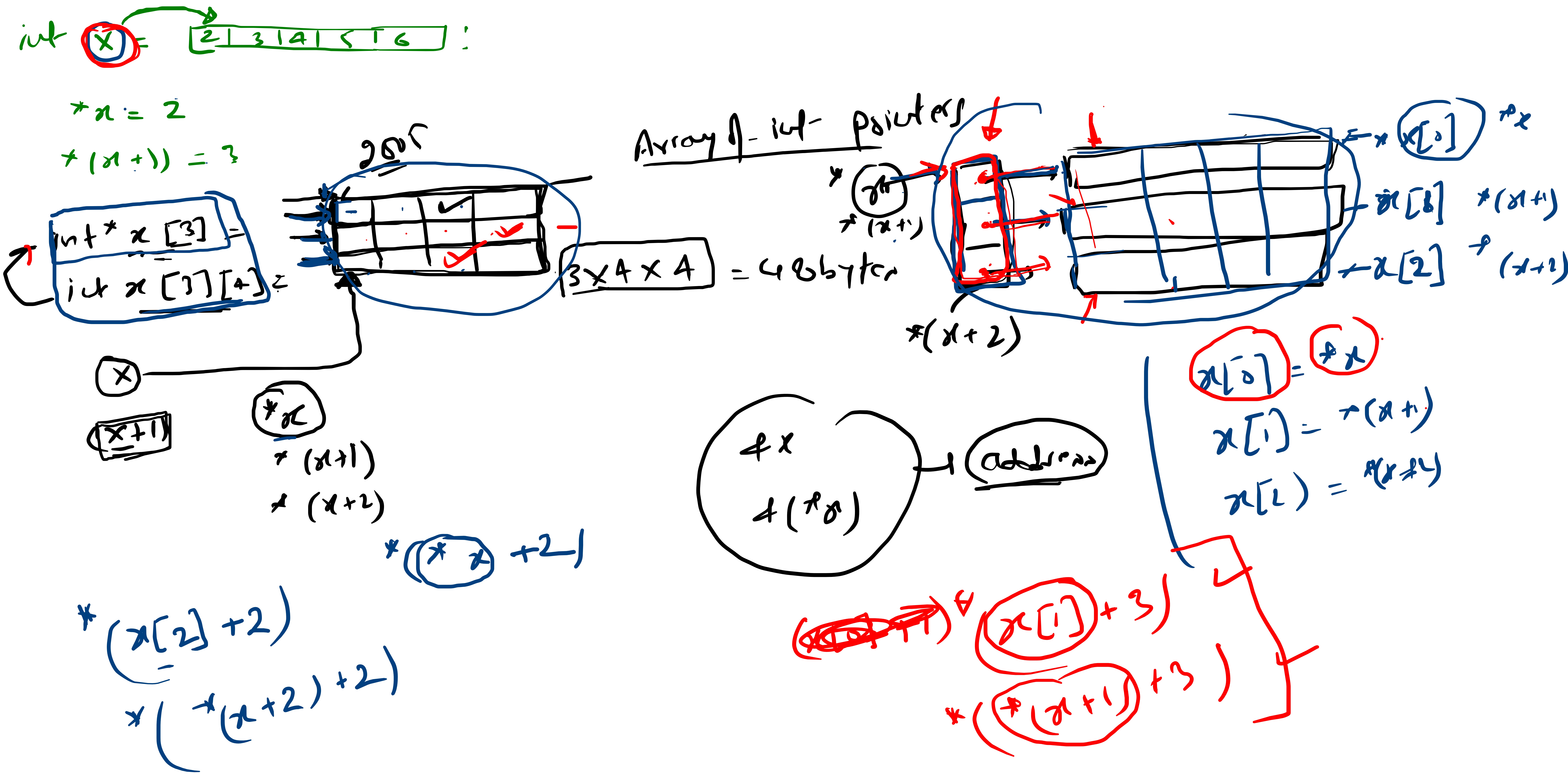

int x = [2|3|4|5|6];
*x = 2
*(x+1) = 3
int* x[3]
int x[3][4]
3x4x4 = 48 bytes
Array of int pointers
x[0] = *x
x[1] = *(x+1)
x[2] = *(x+2)
&x
&(*x)
address
*(x[2]+2)
((x+2)+2)
*(x[1]+3)
((x+1)+3)

Binary Search Tree

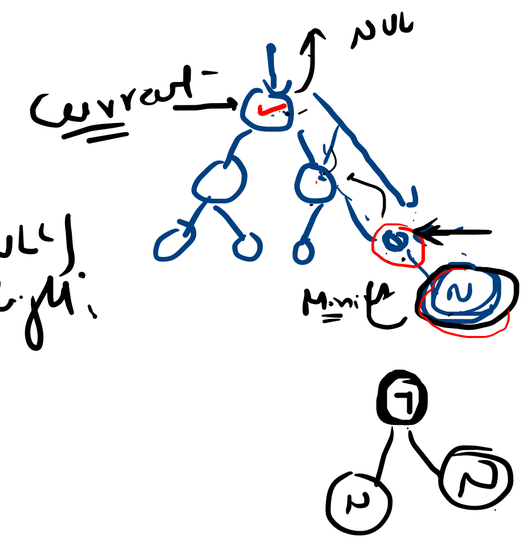
```
Node* Root;
Root = CreateTree();

Node* Current = Root;
Node* Parent = NULL;

while (Current != NULL)
{
    Parent = Current;
    Current = Current -> Right;
}
Print(Parent -> value);
```

```
if (Root == NULL)
    return (Tree is NULL);
```

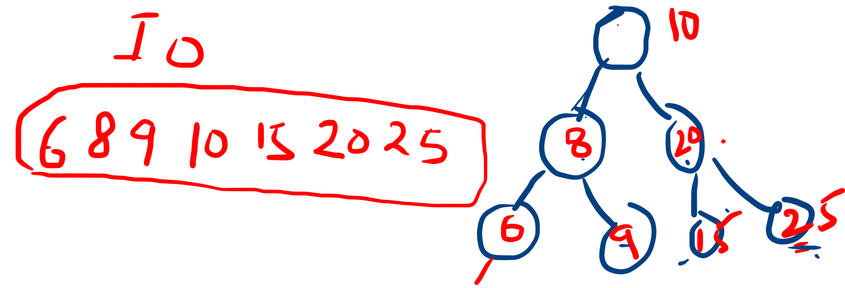
```
while (Current -> Right != NULL)
    Current = Current -> Right;
Print(Current -> value);
```



```
int FindMax(Node* Root)
{
    if (Root == NULL)
        return (MIN_INT);
    else
        return (Max(Root->value, FindMax(Root->R)));
}
```

1. Element Sorting in BST → In order Traversing
   Ascending Order

2. → Descending Order Sorting in BST
   ↳ Reverse In order Traversing



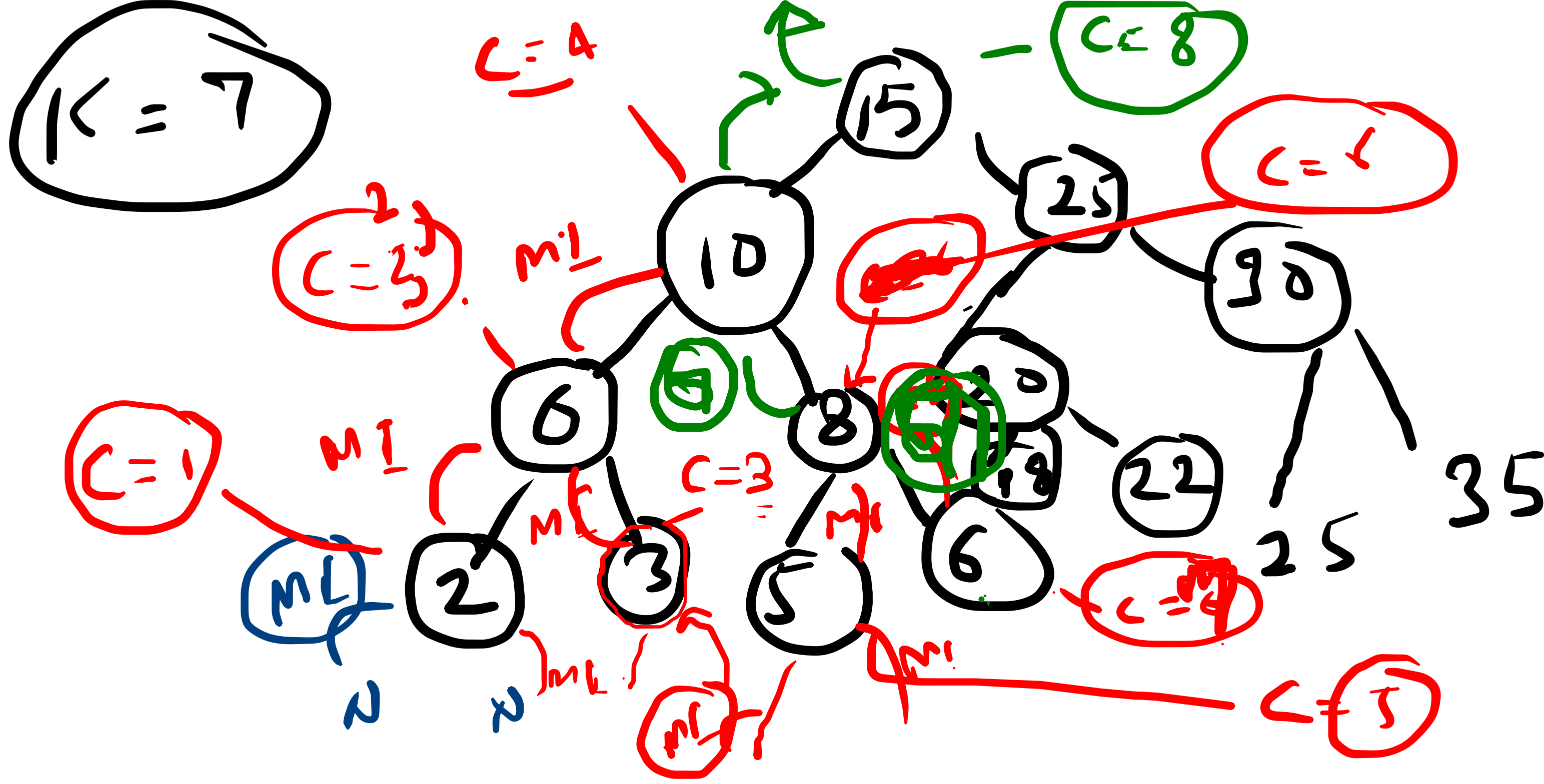
IO
6 8 9 10 15 20 25

IO → L M R

↳ RMO → R M L

25 20 15 10 9 8 6

① Find the k^{th} element in ascending order:



C = 0 ! K = 5.

```
int FindK( Node* Root, int K, int &c)
{
    if ((Root == NULL) || c >= K)
        return (MIN_INT);

    int LV = FindK(Root->Left, K, &c);
    c++;
    ~~if (c == K) return (Root->value);~~
    int RV = FindK(Root->Right, K, &c);
    if (c == K) return (Root->value);
    else return (Min_INT);      return Max(LV, RV)
}
```

Interchanging in case of k^{th} element from Last

① IN order Trav (C=0)

void findK(Node* Root, int K), int C) (K=5)

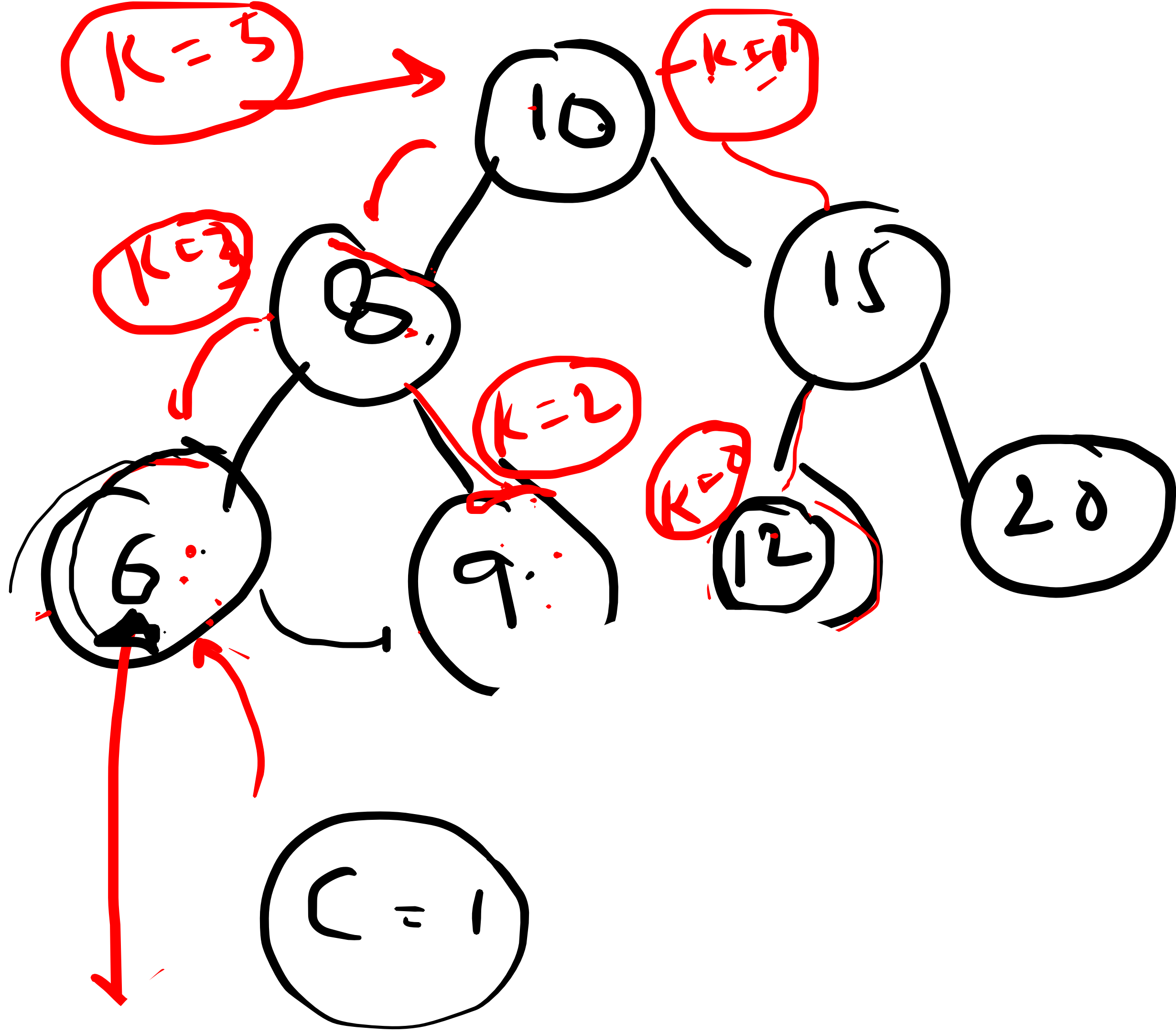
{ if (node == NULL)
return();

else
→ findK(node→left, K, C)

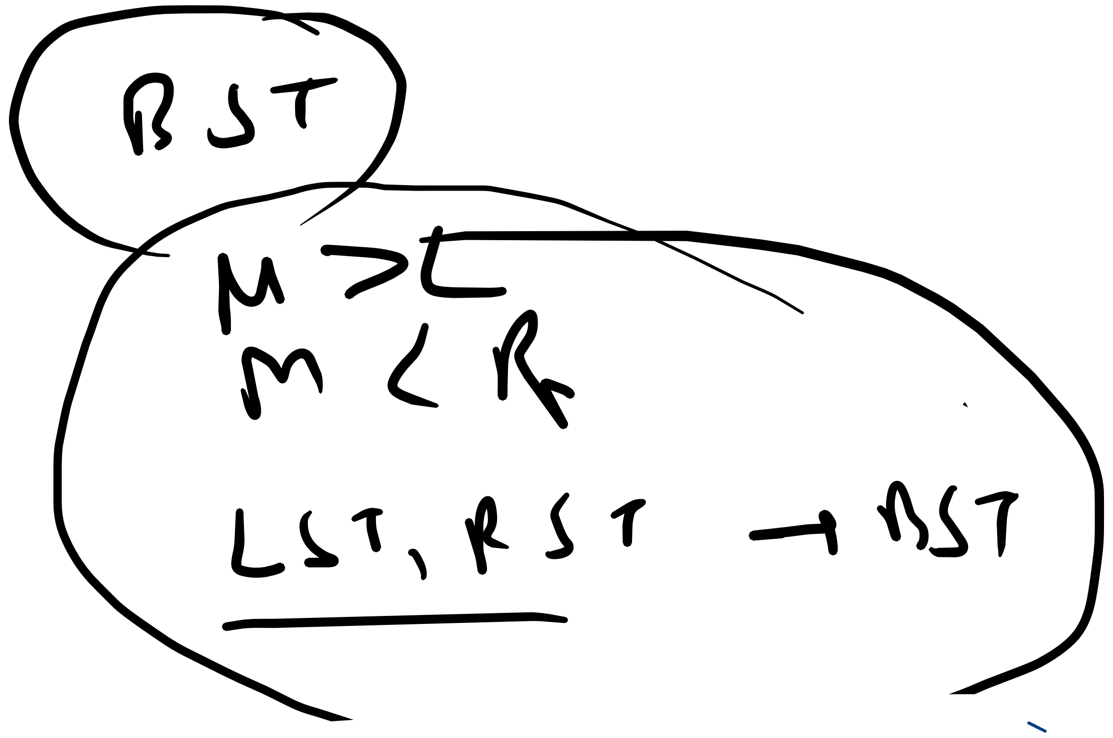
if (C == K)
Print(" Root→value")

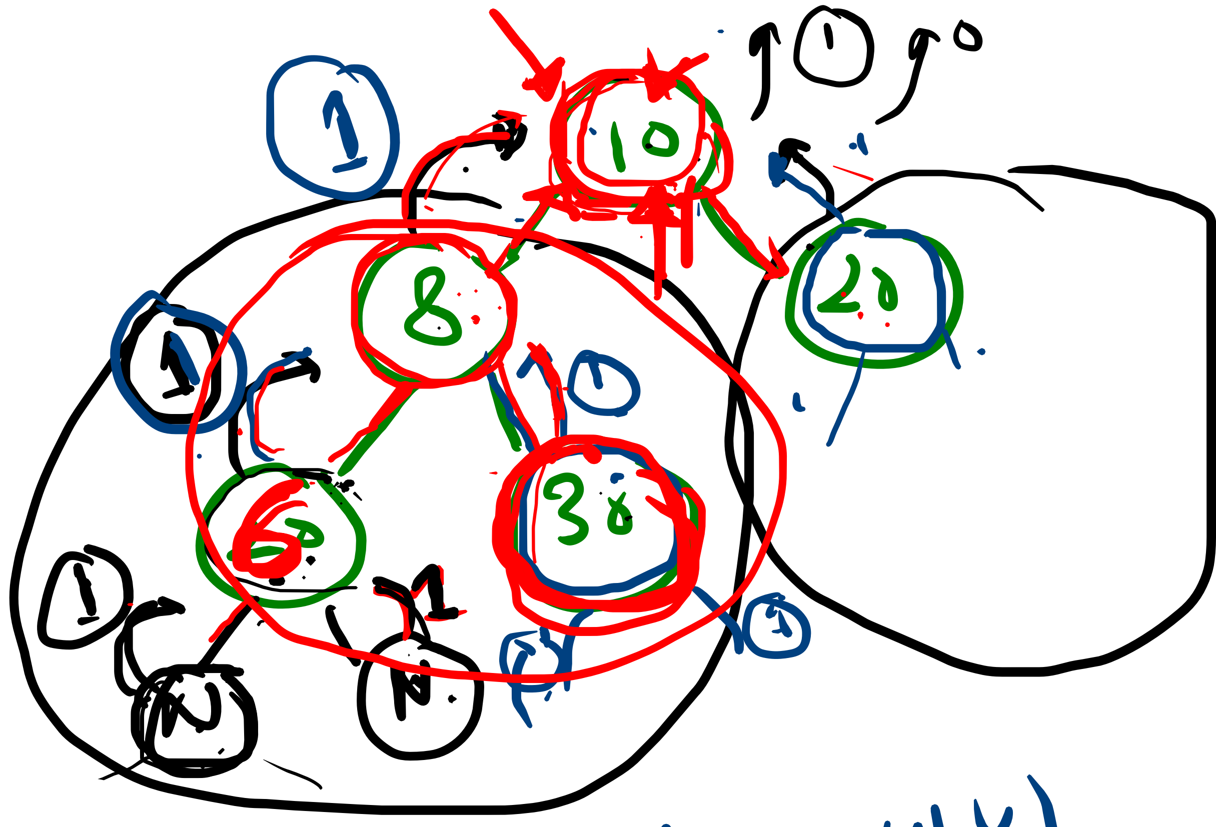
→ findK(Root→right, K, C);

}



① Check Given BT is BST or Not

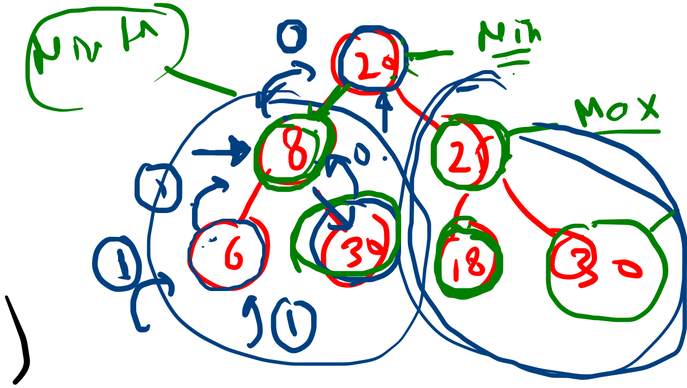





```
int isBST ( Node* Root)
{
    if (Root == NULL)
        return(1);
    else if ( Node→value > Node→Right→value) && Node→Right != NULL)
        return(0);
    else if ( Node→value < Node→left→value) && Node→left != NULL)
        return(0);
    else if (! (isBST( Node→left) && isBST ( Node→Right)))
        return (0);
    else return(1);
}
```

⓪

Min = INT_MIN, Max = INT_MAX;



int isBST (Node * Root, int min, int Max)

{ if (Node == NULL)
return (1);

else if (Node->value > Max || Node->value < Min)
return (0);

else
{ int LV = isBST (Root->left, min, Node->value);
int RV = isBST (Root->right, Node->value, Max);
if(LV != 0 && RV != 0)
return (LV + RV);
else
return (0);
}

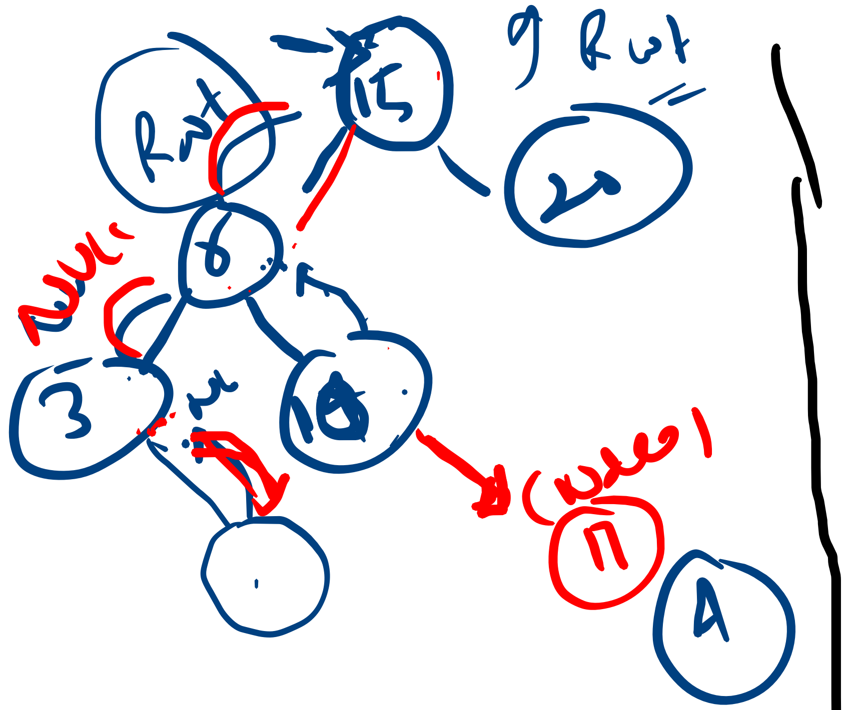
Root	Node	Min	Max
	20	Lmin	Lmx
	8	Lmn	20
1	6	Lmin	8
	30	8	20

① item Search

```
current = Root;
(item)

while (current != NULL)
{
    if (current→value == item)
        return (current);
    else if (current→value < item)
        current = current → Right;
    else
        current = current → left;
}
Print(item not found);
```

(11) (10)



PreOrder

```
Node* search ( Node* Root, int item)
{
    if (Root == NULL)
        return NULL;
    else { if Root→value == item)
        return (Root);
    else if ( Root→value < item)
        Node* x = search (Root → Right, item);
        return (x);
    else
        return (search (Root→left, item));
}
```